SYSTEMS AND METHODS FOR COMPUTER INITIALIZATION

CROSS-REFERENCE TO RELATED APPLICATION

This application is based on a United States provisional application Serial No. 60/180,114, filed on February 3, 2000, which is fully incorporated herein by reference.

BACKGROUND

1. Technical Field:

The present invention relates generally to systems and methods for initializing devices such as computers, computer-based appliances, and processors and, more particularly, to systems and methods for initializing devices comprising one or more volatile logic devices after a power turn-on or a commanded reset.

2. Description of Related Art:

Computers and computer-based appliances, e.g. PCs (personal computers), PDAs (personal digital assistants) and other embedded devices, currently employ large, often complex, operating systems to execute user programs and process data. Conventional operating systems typically range in size from hundreds of kilobytes for, e.g., small PDAs to hundreds of megabytes for, e.g., high-end servers and PCs. As computers and processors become embedded in increasingly more devices, the need for instant computer initialization from a power turn-on (cold boot) or commanded reset (warm boot) becomes even more highly desirable.

Fig. 1 is a block diagram of a conventional computing device illustrating a plurality of boot storage options. More specifically, a plurality of boot devices are shown including a magnetic hard disk 10, CD read only memory (CDROM) 20, floppy disk 30,

network 40 (e.g., wireless network), read only memory (ROM) 50, non-volatile or pre-initialized random access memory (RAM) 60, flash memory 70, or any other non-volatile or pre-initialized volatile memory device (80). Each of the boot devices 10-80 are interconnected by a respective interface 90-160 to their respective data storage/ retrieval adapters 170 - 240. In a typical legacy system as shown in Fig. 1, each storage adapter device 170-240 is connected to an expansion bus 330 via respective interfaces 250-320. The expansion bus 330 is connected to an expansion bus adapter or bridge adapter 340. The expansion bus 330 and expansion bus adapter or bridge adapter 340 are utilized to interconnect the storage adapter devices 170-240 to a main bus 395 of a computer, wherein the computer comprises one or more processors 350, a main RAM storage device 360, a ROM/ flash memory 380, other input/output devices 390 (e.g., mouse keyboard, microphone, etc), and power-up and reset circuitry 370. In this conventional system, the computer reset and power-up circuitry implements both a "cold" and a "warm" boot process as is known in the art.

Fig. 2 comprises a plurality of timing diagrams depicting a conventional "cold" boot process and a conventional "warm" boot process, which are typically implemented in the computer reset and power up circuits 370 (Fig. 1). More specifically, the combination of Figs. 2a and 2b illustrate a cold boot process while Figs. 2c, 2d and 2e illustrate a warm boot process. In Fig 2a, when power is activated, the voltage level increases from an off state to an operating voltage range 420. As further illustrated in Fig. 2a, when the power is reset, a power-up reset operation is performed by momentarily asserting a ***system-reset*** signal 410. Typically, the ***system-reset*** signal 410 tracks the

voltage rise of the power supply, with the exception of a phase lag associated with achieving a power supply voltage threshold 420. The power supply voltage threshold 420 of the ***system-reset*** signal 410 provides a sufficient delay of operation to ensure reliable logic operation is achieved.

In Fig. 2a, when the ***system-reset*** signal 410 reaches threshold voltage 420, the ***system-reset*** signal 410 is asserted active low for a processor reset time period 430 (of **X** milliseconds), during which time an attempt is made to initialize the processor into a known reset state. For example in IBM compatible personal computers, **X** is approximately 200 milliseconds. In Fig. 2b, a system clock preferably begins generating a processor clock signal 440 when (or prior to) the ***system-reset*** signal 410 is asserted to initialize the processor into a known reset state. In Fig. 2a, when the ***system-reset*** signal 410 is deasserted after the time interval 430, the computer's boot device is required to be available either instantaneously or within a predetermined time period thereafter. For example, the industry standard Personal Computer Interface Bus (PCI Bus) Specification Revision 2.2 requires that the boot device be available 5 bus clock cycles after the negation of the bus reset. With a standard PCI Bus clock frequency of 33 megahertz, the boot device must be available in approximately 152 nanoseconds. In particular, this is illustrated in Fig. 2b, wherein the number of clocks for the boot device to be available after the ***system-reset*** signal 410 (as well as signal 415 for the warm boot process) is deasserted is 5 clocks. If the boot device is not available upon the expiration of this predetermined time, the system may crash or be delayed from booting.

Fig. 2c illustrates a warm boot ***commanded-reset-request*** signal 445 that is

generated by a commanded request, e.g., a boot request from the system, an application, operating system, a user, etc. Often computers incorporate a user accessible pushbutton or other switching device by which the user may request a warm boot. Fig. 2d illustrates a ***system-reset*** signal 415 that is generated upon a commanded request. The ***system-reset*** signal 415 is asserted for a processor reset time period 431 (of ***X*** *msec*) upon a commanded reset (similar to Fig, 2a discussed above). It should be noted that signals 410 and 415 are the same signal, except that Figs. 2a and 2d illustrate the different states of the ***system-reset*** signal upon initiation of the cold and warm boot processes. Figure 2e illustrates the system processor clock timing (similar to Figure 2b) for the warm boot process. The boot requirements described above (e.g., processor reset time, time for boot device to be available, etc.) with respect to the cold boot process may be the same as the warm boot process. With the warm boot, it may be assumed that clock signal 440 is operating prior to the ***system-reset*** signal 415 being asserted, although it is not necessary.

Fig. 3a is a flow diagram of the conventional power-up "cold" boot process. A system waits in an idle state for the power supply to be turned on (step 500). When the power is applied (affirmative determination in step 500) and the power supply voltage meets a preset threshold (affirmative determination in step 510), the ***system-reset*** signal is asserted active low for ***X*** *msec* (step 520). At the expiration of the time period ***X***, the ***system-reset*** signal is deasserted (step 530). An optional maximum delay of ***n*** clock cycles (step 540) (or some other prespecified time period) for boot device availability may be inserted. The boot process proceeds to boot the system by loading , e.g., an operating system or application program, etc., if one or more of the associated boot

device(s) are available (step 550).

Fig. 3b is a flow diagram of a conventional "warm" boot process. When an already powered computer receives a ***commanded-reset-request*** signal (affirmative result in step 560), which may be received from a user, the operating system, a computer bridge, an adapter, an application program, etc., the ***system-reset*** signal is asserted for ***X*** *msec* (step 570). At the expiration of time period ***X***, the ***system-reset*** signal is deasserted (step 575). An optional maximum delay of ***n*** clock cycles (step 580) (or some other prespecified time period) for boot device availability may be inserted. The boot process proceeds to boot the system by loading , e.g., an operating system or application program, etc., if one or more of the associated boot device(s) are available (step 590).

A trend within the current art of digital logic devices is the use of volatile programmable logic components, which enable in-situ configuration, and reconfiguration of logic. Logic devices are continually undergoing a technical metamorphosis. Originally, digital designers implemented designs in TTL type logic devices (small-scale integration). These TTL type devices expanded into larger gate counts per package (medium scale integration) with alternate process technologies affording lower power consumption, wider voltage ranges, and specialized electrical interfaces. With the subsequent advent of large and then very large scale integration, embedded systems on a chip became a reality, with dramatically reduced costs. There are, however, various disadvantages associated with implementing dedicated very large scale systems or subsystems on a chip including, but not limited to, high non-recurring design costs, limited flexibility, inherently long fabrication times, high risk due to the time and cost for

design iterations, and perhaps most importantly, lack of field upgradeability.

Consequently, manufacturers of logic devices (such as Xilinx and Alterra) have switched to field programmable logic devices to enable logic users to program devices at the time of logic manufacturer or in the field. To achieve very high logic densities, the current logic devices are volatile and, thus, loose their internal logic program each time power is removed. For example, either a non-volatile memory device or a preloaded volatile memory device must be available for loading the "logic code" into the volatile logic device before the boot code (e.g., operating system, drivers, etc.) is accessible from the boot device. Furthermore, the time needed to load the logic device can be unacceptably longer than the time allotted for a boot device to be available following a system reset, since many volatile field programmable gate arrays utilize serial or parallel loading modes with a relatively low or moderate bandwidths.

Fig. 4 is a block diagram of a conventional system comprising a boot device storage adapter 720 comprising a non-volatile logic device 725. More specifically, the boot device storage adapter 720 employs a non-volatile logic device 725 to access boot data from a boot storage device 700. Data is read from the boot device 700, typically across an industry standard interface 710, via the non-volatile logic device 725, and then onto the optional expansion bus 330, where it read by the computer via the optional expansion bus adapter/ bridge 340. It is to be understood that the boot device 700 represents any of the boot devices 10-80 in Fig. 1.

In the exemplary system of Fig. 4, when the computer issues a ***system-reset-signal*** 410/415 (for a cold boot or warm boot , respectively), and optionally generates an

expansionbus-reset-signal 750, the computer will attempt to load the boot data from the boot storage device 700. If the non-volatile logic device 725 were to be replaced by a volatile logic device, this volatile logic device must first be loaded with the appropriate logic code. Thus one problem associated within the current art is the delay associated with loading a volatile logic boot device after a power-up reset. In the best case this lengthens the boot process that, in many systems, can render the operating system unbootable.

Another limitation within the current art is the need for loading or reloading one or more programmable logic devices 730 upon a warm boot process. Often warm boots are required when a computer application or operating system becomes unstable or corrupted. Depending on the application, however, a warm boot process may or may not require loading of the volatile logic device 725 utilized in the given boot storage adapter or interface 720. These problems within the current art and other limitations are addressed by the present invention.

SUMMARY OF THE INVENTION

The present invention is directed to systems and methods for initializing devices such as computers, computer-based appliances, and processors. In particular, the present invention is directed to systems and methods for initializing devices comprising one or more volatile logic devices after a power turn-on or a commanded reset.

In one aspect of the present invention, a method for initializing a computer system comprises the steps of: sensing a command signal to boot the computer system; generating a first control signal to initialize a boot process; generating a second control

signal to initialize a programmable logic device prior to completion of the initialization of the boot process; and booting the computer system using the initialized programmable logic device.

In another aspect, a boot manager circuit is provided for managing initialization of a computer system. One embodiment of a boot manager circuit comprises: a first sense circuit for sensing power-up and ensuring power stability; a second sense circuit for sensing a command signal to boot the computer system; a control circuit for generating a control signal in response to sensing of a command signal, to initialize a programmable logic device in advance of a boot process; and a state machine for outputting a flag indicative of the type of the type of boot process commanded. Preferably, the first sense circuit is employed to ensure power supply stability prior to generating the control signal for loading the programmable logic device.

In another aspect, the boot manager circuit processes an external reset request to warm boot the system either with or without reloading the programmable logic device.

In yet another aspect of the present invention, the boot manager circuit is employed to simultaneously or sequentially load multiple programmable logic devices depending on the desired application.

In another aspect of the present invention, a system for initializing a computer comprises: a boot storage device for storing initialization program code for initializing a computer during a boot process; and a boot device adapter, operatively interfaced with the boot storage device, for accessing the initialization program code from the boot storage device in response to a request from the computer system; wherein the boot

device adapter comprises: a programmable logic device; and a boot control circuit for generating a control signal to initialize the programmable logic device in advance of the boot process.

The computer initialization system may further comprise a memory device for storing logic code associated with the programmable logic device. The memory device preferably comprises non-volatile memory residing on the boot device adapter, the computer system, device, appliance, or the boot device. In response to the control signal, the programmable logic device can self-load the logic code from the memory device.

In another aspect, the computer initialization system comprises a digital signal processor (DSP) that initializes the programmable logic device in response to the control signal. The DSP preferably resides on the boot storage device adapter. The DSP can be connected to the programmable logic device through a dedicated bus of the DSP or a common bus. The DSP retrieves logic code associated with the programmable logic device from a memory device residing on the boot storage device, the boot device adapter, and/or the computer system. In addition, the memory device may be employed to store logic code associated with the DSP.

In another aspect, the boot device may be employed to load the logic code of a programmable logic device that is located in the boot device itself, in the boot device adapter or in the PC or appliance.

These and other aspects, features and advantages of the present invention will become apparent from the following detailed description of preferred embodiments, which is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of conventional computer system comprising a plurality of representative boot devices;

Figs. 2(a)-2(e) comprise timing diagrams that illustrate conventional "cold" and "warm" boot processes;

Figs. 3a and 3b comprise flow diagrams respectively illustrating a conventional "cold" boot process and "warm" boot processes;

Fig. 4 is a block diagram of a conventional initialization system comprising a boot device storage adapter comprising a non-volatile logic device;

Fig. 5 is a high-level block diagram of a boot management circuit according to an embodiment of the present invention;

Fig. 6 illustrates timing diagrams of a boot process according to one aspect of the present invention implementing the boot management circuit of Fig. 5 in a legacy computer system;

Fig. 7 illustrates timing diagrams of a boot process according to one aspect of the present invention implementing the boot management circuit of Fig. 5 in a non-legacy computer system;

Figs. 8a and 8b respectively comprise flow diagrams of a "cold" and "warm" boot process according to one aspect of the present invention;

Fig. 9 is a schematic diagram of a boot management circuit according to an embodiment of the present invention;

Fig. 10 is a timing diagram illustrating the waveforms at various nodes in the diagram of Fig. 9;

Fig. 11 is a block diagram of an initialization system according to an embodiment of present invention utilizing a boot management circuit;

Fig. 12 is a block diagram of an initialization system according to another embodiment of present invention;

Fig. 13 is a block diagram of an initialization system according to another embodiment of present invention;

Fig. 14 is a block diagram of an initialization system according to another embodiment of present invention;

Fig. 15 is a block diagram of an initialization system according to another embodiment of present invention; and

Fig. 16 is a block diagram of an initialization system according to another embodiment of present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

The present invention is directed to systems and methods for initializing devices such as computers, computer appliances, and processors. More specifically, the present invention is directed to system and methods for initializing (programming) a volatile programmable logic device (employed in the computer or processor boot process) in advance of its application. In the following description, it is to be understood that system elements having equivalent or similar functionality are designated with the same

reference numerals in the Figures. The systems and methods described herein may be implemented in various forms of hardware, software, firmware, or a combination thereof.. Preferably, the present invention is implemented utilizing a combination of novel analog signal processing techniques and digital logic applied to programmable logic technologies, high density non-volatile storage devices, digital signal processors or any other type of processing device or technique. Additionally, the present invention is applicable to any type of programmable logic device utilized in the boot process of a computer or appliance, which is utilized to load the operating system, drivers, application code, or any other software and firmware.

Fig. 5 illustrates a high-level block diagram of a boot management circuit 1070 according to an embodiment of the present invention. In general, a preferred architecture of the boot management circuit 1070 comprises a component for sensing power-up and user/system commanded reset requests and a component for generating control signals in response to such power-up and commanded requests for causing logic code to be loaded into one or more programmable logic devices in advance of use of such programmable logic devices in a boot process. Furthermore, a preferred boot management circuit comprises a component for determining the type of boot process that was initiated (e.g., power-up or commanded reset) and providing an indication of such boot process.

The exemplary boot management circuit 1070 of Fig. 5 comprises a plurality of inputs for receiving a ***power-supply-monitoring*** signal 400, an optional ***system-cold-boot-reset-request*** signal 740, an optional ***commanded-warm-boot-reset-request*** signal 760, (optionally) a legacy ***system-reset*** signal 410/415; and optionally an ***expansionbus-***

reset-signal 750.

In future system implementations (i.e. non-legacy) utilizing the present invention, a preferred implementation is to employ either the ***power-supply-monitoring*** signal 400 and/or the ***system--cold-boot-reset-request*** signal 740 for determining cold boot requests. Similarly the ***commanded-warm-boot-reset-request*** signal 760 is utilized to sense warm boot requests.

In existing (i.e., legacy) computer architectures, signals such as 740 and 760 are typically not available from the current hardware/software implementation, thus only existent signals must be utilized that are generated by the computer or expansion bus. Hence, preferably, for implementations within a legacy system, the ***power-supply-monitoring*** signal 400 coupled to either the power supply or the expansion bus power is utilized in conjunction with either an ***expansionbus-bus-reset*** signal 750 or the ***system-reset*** signal 410/415 to distinguish between a cold and warm boot. It should be noted that in legacy systems, the ***expansionbus-bus-reset*** signal 750 is typically derived from the ***system-reset*** signal 410/415, usually with appropriate buffering or some other form of signal conditioning. Thus, when the boot management circuit 1070 is implemented in a legacy system, the ***system-reset*** signal 410/415 or ***expansionbus-bus-reset*** signal 750 is preferably monitored along with ***power-supply-monitoring*** signal 400. If the ***system-reset*** signal 410/415 or the ***expansionbus-bus-reset*** signal 750 is asserted and the ***power-supply-monitoring*** signal 400 is above the specified threshold 420 (deemed power supply valid), then the boot process is a warm boot. However, if the ***system-reset*** signal 410/415 or the ***expansionbus-bus-reset*** signal 750 is asserted and the ***power-supply-monitoring***

signal 400 is below the specified threshold 420 (deemed power supply invalid), then the boot process is a cold boot.

The exemplary boot management circuit 1070 further comprises a plurality of output pins for outputting a ***boot-device-reset*** signal 1030, a ***system-reset*** signal 410/415 (depending if a cold or warm boot process is initiated), and (optionally) indicator signals such as a ***warm-boot-flag*** indicator 1050 and/or a ***cold-boot-flag*** indicator 1040.

For non-legacy systems, it may be advantageous to generate the optional ***system-reset*** signal 410/415 signal from the logical OR of the ***system-cold-boot-system-reset-request*** 740 and the ***commanded-warm-boot-reset-request*** 760 to minimize logic circuitry. In the case of non-legacy systems the ***system-reset*** 410/415 is not an input to the boot management circuit 1070 but is an output generated by the boot management circuit 1070. The non-legacy use of the ***system-reset*** 410/415 is similar to its use in legacy systems where the signal 410/415 may be utilized to reset processors, peripherals, and all other non-volatile elements of the boot device. For non-legacy systems the ***warm-boot-flag*** signal 1050 and the ***cold-boot flag-signal*** 1040 may be derived from their respective input request signals 760 and 740. For legacy systems the previously specified logic is preferably applied.

It is to be appreciated that the boot management circuit 1070 according to the present invention can be readily employed in all existing and future platform architectures. As such, it is to be further understood that the input signals 400, 740, 760, 410/415 and 750 and output signals 1030, 410/415, 1050 and 1040 are labeled with names that describe their associated functions, and that, depending on the platform,

system or application, other signals providing corresponding functions of the signals depicted in Fig. 5 may be applied.

It is to be further appreciated that the ***boot-device-reset*** output 1030 provides a mechanism for initiating the loading of the programmable logic device in advance of the use of the programmable logic device in the boot process. The ***cold-boot-flag*** output signal 1040, and the ***warm-boot-flag*** output signal 1050 indicate which type boot process has been commanded. Advantageously, the output flags 1040, 1050 provide an indication that allows additional logic that is implemented in either hardware, software, or any combination thereof, to elect to either load or reload program code in the programmable logic device, or leave intact the current programming of the programmable logic device. Indeed, depending on the application, it may or may not be advantageous to reload the programmable logic devices. The ***system-reset*** output 410/415, typically resets either all or a portion of the PC components, appliance components, or system components.

It should be noted that while singular input signals are shown, the input signals are indicative of a signal type and may each occur in pluralities. For example, the ***power-supply-monitoring*** signal 400 may comprise a plurality of power senses. Typically, the individual senses that may be utilized include +5 volts, +3.3 volts, +1.8volts, +1.5 volts, +12 volts, and -12 volts, although any suitable voltage or combination thereof may be implemented. The ***power-supply-monitoring*** signal 400 may be monitored and logically combined using any suitable conventional logic equation to provide a power sense. Certain power supply voltages are used for specific functions. For example +5 and +3.3 volts may be utilized for logic input and output while lower voltages such as +1.5 volts

may be utilized for internal logic cores of high-density logic devices and processors. The process of monitoring voltages is application specific to a number of parameters including the specific logic implementation, system functionality, and/or processors utilized in the computer or appliance. Further, the ordering may be sensed to ensure that the power has been applied in the proper sequence, i.e., the logic cores are powered before I/O devices to avoid latchup).

Referring now to Fig. 6, a plurality of timing diagrams are shown depicting both cold and warm boot process for implementation of the boot management circuit 1070 in a legacy system. In Fig. 6(a), upon power-up of the system, a ***power-supply-monitoring*** signal 400 reaches a stable and acceptable voltage operating range 405 within a given time period. The phase lag associated with the ***power-supply-monitoring*** signal 400 may vary based upon factors such as the source impedance and load condition of the input source and the internal functioning of the power supply, topology, analog or switching, frequency of switching, capacity, etc. Then, as illustrated in Fig. 6(b) (which is similar to Fig. 2(a)), the ***system-reset*** signal 410 for a cold boot is generated, which typically approximates the ***power-supply-monitoring*** signal 400 until a minimum threshold voltage 420 is reached so as to enable proper logic operation. Once the threshold 420 is reached, (with a cold boot process), the ***system-reset*** signal 410 is asserted for ***X*** milliseconds. Although the time period ***X*** is typically 200 milliseconds, any suitable value for ***X*** may be implemented in the present invention, as the present invention is not specific to any given value. Indeed, it is anticipated that this time period with grow smaller in future systems. Additionally, it is not required that the ***system-reset*** signal 410

follow the ***power-supply-monitoring*** signal 400. Indeed, the ***system- reset*** signal 410 may remain continuously asserted until it is negated.

For a warm boot process, in Fig. 6(c)(which is similar to Fig. 2(d), the ***system-reset*** signal 415 is asserted for ***Y*** milliseconds, wherein ***Y*** may be any suitable time period. It is to be understood that notwithstanding that Fig. 7 illustrates that $X = Y$, their values may be different depending on the application. For instance, warm boot processes may benefit from a reduced time ***Y*** since certain system elements may not require initialization or since there are less transients within a system that is already powered (pre-charged capacitances and inductances).

In Fig. 6(d) (which is similar to Fig. 2(b)), the system processor clock generates the clock signal 440, which typically begins operation once the input power has reached an acceptable operating level. While it is possible that the system clock is held in reset until the ***system-reset*** signal 410/415 is negated, it is often preferable to allow the system clock 440 to operate during the time interval ***X***, (or ***Y***) to aid the system logic and/or initializing the processor into a known state. For example, many general-purpose processors and digital signal processors typically require a number of input clocks to properly operate when released from the reset state. It is to be understood that the system processor clock signal 440 is shown for pedagogical purposes and is not a required element of the present invention. Again, a maximum delay from reset negation to boot device availability may be implemented based on, e.g., a maximum number ***n*** of clock cycles (or some other prespecified time interval). For example, in the Personal Computer Interface Specification Revision 2.2, the boot device must be available 5 clock cycles

after the negation of reset on the bus, or the system may crash. At the current system clock rates of 33 and 66 megahertz, this approximately corresponds to maximum times of 150 and 75 nanoseconds, respectively. Again, the maximum delay from reset negation to boot device availability may be anywhere from zero to some maximum specified time value delineated in any convenient units including multiples of clock periods.

As further illustrated in Fig. 6(e), in a cold boot process, the ***boot-device-reset*** signal 1030 preferably follows the ***power-supply-monitoring*** signal 400, although this is not essential for implementing a boot process according to this invention. With a warm boot process, the ***boot-device-reset*** signal 1030 will change from a negated state to an asserted state upon the system-reset-signal 415 being asserted. When the ***system-reset*** signal 410 is asserted to active low, the ***boot-device-reset*** signal 1030 is asserted to active low for ***Z*** *msec,* where ***Z*** comprises a number that is shorter than the respective ***X*** or ***Y*** time periods for cold and warm boots respectively. In general, the period of time ***X - Z*** (or **Y - Z** with a warm boot) should be sufficient to load and activate the programmable logic device elements necessary for boot, which may comprise of a portion of one programmable logic device or one or more programmable logic devices. This time period may be shortened, for instance when reloading of the volatile logic device is not necessary.

Further, an optional internal state machine is set to provide an indication of the mechanism that is responsible for initiating the boot process. Preferably, this mechanism comprises either asserting or negating the ***cold-boot-flag*** 1040 along with the complementary ***warm-boot-flag*** 1050. In particular, as shown in Figs. 6(f) and (g), in a

preferred embodiment, the signals 1040, 1050 change state when the ***boot-device-reset*** signal 1030 is negated. It is to be appreciated that in other embodiments of the present invention, the state machine and indicator signals may be valid earlier than negation of the ***boot-device-reset*** signal 1030, if so desired.

Referring now to Fig. 7, a plurality of timing diagrams are shown depicting both cold and warm boot process for implementation of the boot management circuit 1070 in a non-legacy system. These timing diagrams are similar to the corresponding timing diagrams of Fig. 6, expect that in Fig. 7, the ***system-cold-boot-reset-request*** signal 740 in Fig. 7(e) and the ***commanded-warm-boot-reset-request*** 760 signal in Fig. 7(f) are utilized directly as the cold and warm boot requests, as opposed to their derivation in legacy applications as described herein.

Referring now to Figs. 8a and 8b, flow diagrams respectfully illustrate a "cold" and "warm" boot process according to one aspect of the present invention. More specifically, with a preferred cold boot process as illustrated in Figure 8a, a test or other check is continuously performed to determine whether the power supply has been turned on (step 500). If power has been applied (affirmative determination in step 500), a determination is then made as to whether as to whether the voltage, current, and/or aggregate power from one or more supply voltages has met a predetermined threshold (step 510). Furthermore, in the case of underdamped or critically damped supplies, it may also be desirable to test both high and low voltage rails to ensure that the power supply has settled. This process may also include a time delay to ensure that the supply is not in the process of oscillating above and below the thresholds, inducing a false power

supply valid indication. Multiple senses (and the order thereof) may be combined to derive the most reliable or otherwise optimal indication of power supply validity, if so desired.

Upon power supply validity (affirmative determination in step 510), preferably, two parallel processes occur (i.e., a first parallel process comprising steps 520, 530 and 540, and a second parallel process comprising steps 1200, 1210, 1220 and 1230). More specifically, the first parallel process initiates with the ***system-reset*** signal 410 being asserted for ***X*** msec (step 520), or some other prespecified time period. Then, after expiration of the time period ***X***, the ***system-reset*** signal 410 is deasserted (step 530). Then, a time delay 540 (shown in Fig. 7(n) as ***n*** clock cycles) is optionally inserted before the boot process 550 begins.

In the second parallel process, the ***boot-device-reset*** signal 1030 is asserted (substantially concurrently with the assertion of the ***system-reset*** signal 410) for ***Z*** *msec* (step 1200), wherein ***Z < X.*** During the time period **Z**, any logic devices utilized on the boot storage adapter or processors may be reset including those utilized to load the volatile logic device including the boot storage device. At the expiration of ***Z***, the ***boot-device-reset*** signal 1030 is deasserted (step 1210). Then, the boot state indicator is optionally read to ensure the appropriate boot process and loading of the correct logic code into the programmable logic device (step 1220). Next, the programmable logic device is loaded (step 1230). It is to be noted that a delay may then be encountered waiting for completion of the first parallel process. Finally, the boot process begins (step 550).

With a preferred "warm" boot process as depicted in Fig. 8b, a test or other check is continuously performed to determine whether a request for reset (e.g., user, system, application, etc.) has been asserted (step 560). If so (affirmative determination in step 560), two parallel processes occur (a first parallel process comprising steps 520, 530, and 580, and a second parallel process comprising steps 1200, 1210, 1220, and 1230). The first parallel process initiates with ***system-reset*** signal 415 (master reset) being asserted active low for time period of ***Y*** *msec* (step 520), wherein ***Y*** may be equal to any suitable time period. After the ***system-reset*** signal 415 has been asserted for the prespecified time period ***Y***, the ***system-reset*** signal 415 is deasserted (step 530) and a maximum time delay of ***n*** clock cycles is optionally mandated (step 580) for the availability of the boot device. The boot process begins (step 590) if the boot device is available.

In the second parallel processes, the ***boot-device-reset*** signal 1030 is asserted for ***Z*** *msec* (step 1200), wherein ***Z*** < ***Y***. After the expiration of ***Y***, the ***boot-device-reset*** signal 1030 is deasserted (step 1210). Then, the boot state indicator is optionally read to ensure the appropriate boot process and loading of the correct logic code into the programmable logic device (step 1220). Next, the programmable logic device is loaded (step 1230). It is to be noted that a delay may then be encountered waiting for completion of the first process. Finally, the boot process begins (step 590).

Fig. 9 illustrates a schematic circuit diagram of a boot management circuit 1070 (Fig. 5) according to a preferred embodiment of the present invention. In the illustrated embodiment, the boot management circuit 1070 comprises a power supply sense circuit 1071 comprising an input node A, a diode D1, resistors R1 and R4, a capacitor C1, a

jumper J1 and output node B. Preferably, R1 is at least one order of magnitude lower than the resistance R4. The power sense circuit 1071 is essentially a dual time constant integrator with R4 and C1 defining the charge time constant and R1 and C1 defining the discharge time constant. While it is preferable that D1 have a low forward bias voltage such as those found in Schottky diodes, it is not necessary as R4 bleeds out residual charge at a low rate for voltages below and the forward bias of the diode D1 when the power supply sense voltage (at node A) drops below the forward bias of the diode D1. The jumper J1 may be used to short-circuit the other components of the circuit 1071 to eliminate the effect of the charge/discharge time constants in the event that the power supply provides the needed characteristics.

The boot management circuit 1070 further comprises a Schmidt inverter U1, operatively connected to the power sensor circuit 1071 at node B, and operatively connected to an input of a falling edge differentiator 1072 at node C. The falling edge differentiator 1072 preferably comprises a capacitor C2 and resistor R2. The inverter U1 is employed in conjunction with the falling edge differentiator 1072 to generate a pulse (at node D) that is utilized to create a ***boot-device-reset*** signal 1030 (at node E) by means of a two input Schmidt AND gate U2 (which operates as an active low in, active low out, OR gate).

The boot manager 1070 further comprises a system reset circuit 1073 comprising a jumper J2 and resistor R5, for processing the legacy ***system-reset*** signal 410/415, a jumper J3 again with resistor R5, for processing the ***expansionbus-bus-rest*** signal 750, and a jumper J4 and resistor R6 for processing the ***commanded-warm-boot-reset-request***

signal 760 (which are discussed above with reference to Fig. 5). By inclusion of jumpers J2, J3, or J4, and an appropriately asserted *reset-request* signal at the respective input, a ***boot-device-reset*** signal 1030 (at node E) will be generated. It should be noted that each of the jumpers J1-J4 illustrated in Fig. 9 may comprise any suitable conventional switching device, including, but not limited to, passive or active, or static or dynamic, switching devices. When jumpers J2, J3 and J4 are not installed, resistors R5 and R6 are utilized as logic pull-ups to default the system and external reset requests to the inactive state.

Next, an AND gate U3 is utilized to combine the system and user reset request signals into one combined request (output at node H). A differentiator circuit 1074, comprising a capacitor C3 and a resistor R3, acts as a rising edge differentiator whose output (at node I) is then supplied to a Schmidt inverter U4 for waveshaping the reset request pulse (output to node J). It is to be noted that the values of C3 and R3 are preferably selected based on the desired pulse width time of the signal output at node J. U2 is again utilized to logically ***OR*** the power-up system reset, and external reset request. An S/R flip-flop U6 implements a state machine that asserts and negates the appropriate warm-boot and cold boot flags, 1050, 1040. A Schmidt inverter U5 is utilized to convert the polarity of the power-up reset for appropriate operation of the state machine. As previously noted, the ability for a computer or computer appliance to inquire about the boot initiator is advantageous, as it provides a mechanism for rebooting of only those system and reloading of those programmable logic devices that are required or desired.

Figure 10 comprises a plurality of timing diagrams illustrating the state of the

signals located at each node (A-J) of the boot manager circuit illustrated in Fig. 9. More specifically, the letter designations **A-J** correspond to the waveforms generated as a function of time at the associated nodes labeled in the schematic diagram of Fig. 9. Signal **A** illustrates a typical turn-on transient of a power supply of a computer or computer appliance. With computers comprising low cost switching power supplies that operate with the power consumption of hundred of watts, this time is typically 5 *msecs.* This time and the actual profile vary significantly based on the design and individual system tolerances. It is expected that as power supplies continue to become more efficient and faster, the switching frequencies will increase causing a shorter turn on profile. The present embodiment does not rely on any specified time or wave shape for the turn on profile.

Further, Signal **B** depicts the voltage waveform input to U1 as generated by Signal **A** (power supply sense input) charging the C1 with time constant R4C1. The Schmidt trigger U1 provides threshold hysteresis, affording a higher level of nose immunity. It is to be noted that U1 is not a necessary component of the invention.

Signal **C** is the output of the Schmidt trigger inverter U1, and Signal D is the output as processed the differentiator circuit 1072. Signal D is input to the Schmidt AND gate U2 to generate the ***boot-device-reset*** signal 1030 as illustrated Signal E. Signal D is also input to the Schmidt inverter U5, which outputs Signal K. Signal K is input to S/R flip-flop U6 which then sets the indicator ***cold-boot-flag*** 1040 shown as Signal L. The complementary output indicator ***warm-boot-flag*** 1050 is simultaneously negated as shown Signal M.

The ***system-reset*** signal 410/415 and ***commanded-warm-boot-reset-request*** signal 760 are shown disabled by the removal of jumpers J2 and J4, respectively. Both requests are shown active low and Signals F OR G show individual requests. The Schmidt AND gate U3 logically *OR*s the request and generates Signal H as an input to the rising edge differentiator circuit 1074. The output of the differentiator circuit 1074, Signal I, is the input to Schmidt inverter U4 which generates signal J that is input to both U2 and U6.

A ***boot-device-reset*** signal 1030 is then generated and the S/R flip-flop U6 is set to the ***cold-boot-flag*** 1050 as indicated by Signal **M**. As before, the use of Schmidt function provides additional noise margin through the logic's hysterisis and is optional to all logic functions in the present invention.

Fig. 11 is a block diagram of a boot management system according to an embodiment of present invention utilizing a local non-volatile storage device and volatile logic device. A boot storage device 700 stores an operating system and/or application programs, which are to be loaded upon system initialization. The boot storage device 700 may comprise any mass storage device. An optional interface 710 operatively connects the boot storage device 700 to a boot device storage adapter 770 (although the boot storage device may be directly connected to a computer, computer appliance, processor, etc). The boot device storage adapter 770 comprises a volatile logic device 730 that is implemented to access the boot storage device 700 and/or process operation of the boot storage device 700. The boot device storage adapter 770 further comprises a non-volatile logic device 736,which stores the logic program for the volatile logic device 730. As indicated above with reference to Fig. 5, a boot management circuit 1070 (incorporating

one embodiment of the present invention) receives boot requests from a ***power-supply-monitoring*** signal 400, an optional ***system-cold-boot-reset-request*** signal 740, a ***commanded-warm-boot-reset-request*** signal 760, (optionally) a legacy ***system-reset*** signal 410/415, and (optionally) an ***expansionbus-reset*** signal 750.

Advantageously, as indicated above, the boot management circuit 1070 ensures that the volatile logic device 730 is preprogrammed and operational prior to a request from the external adapter/bridge 340 (or the PC/appliance itself) to access, or receive data from the boot storage device 700. It is to be appreciated that as explained above, the boot management circuit 1070 is readily backwards compatible with legacy systems because the boot management processes and converts boot signals of legacy systems into signals that are used for implementing the booting techniques described herein. It is to be further appreciated that the methods and systems described herein may be implemented in legacy free systems.

The volatile logic device 730 may utilize a conventional self-loading mechanism to load its associated logic code in response to a control signal from the boot management circuit Alternatively, the logic code for the volatile logic device can be loaded by means of external logic circuitry. It should be noted that within the current art, boot storage devices may be addressed as memory or mass storage devices. Further, the access to the boot storage device 700 may be as a slave or the device 700 may be a master initiating the transfer itself.

Further it should be noted that in Fig. 11 the ***power-supply-monitoring-signal*** 400 is shown operatively sensing one or more voltages from the expansion bus as is typical in

legacy implementations. This signal may also be coupled to the power supply as in non-legacy implementations.

Fig. 12 is a block diagram of a boot management system according to another embodiment of the present invention, wherein a boot device storage adapter 771 comprises a digital signal processor or any other processor 1320 and (optionally) a RAM 1330. In this embodiment, the non-volatile memory device 736 may store the program(s) for the volatile logic device 730 and the programs for the digital signal processor(s) 1320. Also, the digital signal processor 1320, volatile logic device 730, and non-volatile memory device 736 and RAM 1330 may be operatively connected to the volatile logic device 730 via a common bus structure 1340 (although they may be connected via dedicated pathways, or any combination of dedicated and common buses). The non-volatile memory 736 may be contained within the digital signal processor 1320 or any other element of the current embodiment, i.e., the volatile logic device 730 itself, bus adapter 340, or frontside bus 395. To load the volatile logic device 730, the device 730 may utilize a conventional self-loading mechanism, external logic circuitry, or the digital signal processor 1320 via a common or dedicated pathway.

Fig. 13 is a block diagram of a boot management system according to another embodiment of the present invention. The system includes of one or more non-volatile devices 795, 785, 796 that are remotely located from the boot device adapter 772.. For example, as specifically illustrated in Fig. 13, a non-volatile memory device 795 may be remotely located on the external expansion bus 330 of the expansion bus adapter 340. In addition to, or alternatively, a non-volatile memory device 785 may be remotely located

on the local or frontend bus 395. It is to be appreciated that a non-volatile memory device 795 and/or 785 may be located in multiple locations either in part, or in whole, and each non-volatile memory device 795 and/or 785 may be dedicated to the programmable logic device 730 or shared for other functions. Further, it is to be appreciated that the non-volatile logic code may be located within the boot storage device itself 796 and bootstrap loaded into the volatile logic device 730. In this embodiment, the boot device 700 may be utilized to load the appropriate programming code into a programmable logic device 730 located in the boot device adapter 772, or a programmable logic device located in the boot device 700 itself, or in the PC or computer appliance.

Fig. 14 is a block diagram of a boot management system according to another embodiment of the present invention. The architecture of the boot device storage adapter 773 comprises a digital signal processor 1320 (or other processor) having a dedicated input/output bus 1350 that is utilized for programmable device loading. The addition of one or more digital signal processors 1320 or other processor is utilized on the boot storage device adapter 773 with optional RAM 1330. The non-volatile memory device 736 may store both the volatile logic device program, and programs for the for the digital signal processor(s) 1320.

A dedicated pathway 1350 (parallel programming bus) is shown between the digital signal processor(s) 1320 to the volatile logic device 730, which may be used for express loading the volatile logic device 730. A common bus 1340 is utilized for accessing the non-volatile memory device 736. In the case of a DSP 1320, bus 1340 is often referred to as the main bus and bus 1350 the I/O or expansion bus. In this

embodiment, any combination of dedicated/common busses is permissible, depending on the application. Further, the digital signal processor 1320 may share the dedicated input/output bus 1350 for any purpose including special purpose functions. For example, the Texas Instruments C62x and C64x family of digital signal processors has a dedicated bus for connection to one or more industry standard T1/E1 telecommunications ports. The signals transmitted on bus 1350 in their current format, or when enabled for general purpose I/O, may be utilized for programming the non-volatile memory device 736. Indeed, both data and strobe signals may be generated along with feedback for reading programming status as required. By using any suitable multiplexing techniques, the port may also be used for T1/E1 functions, thereby saving logic and cost.

A translation software program may be utilized to orient the logic program for optimal storage and access in the non-volatile memory device 736, thereby saving processor cycles and minimizing the time for programming the volatile logic device 730. The translation program precomputes the optimal storage patterns for the volatile logic program. The non-volatile memory device 736 may be stored in the digital signal processor 1320, or any other element such as the volatile logic device 730, bus adapter 340 located on frontside bus 395.

Fig. 15 is a block diagram of a boot management system according to another embodiment of the present invention comprising a boot storage device adapter 774 comprising a plurality of programmable volatile logic devices 1350, 1360. The non-volatile memory device 736 may be utilized to store the programs of one or more volatile memory devices 1350, 1360. It should be noted that although two volatile memory

devices are shown, the system may employ more than two volatile memory devices. In addition, although one non-volatile memory device 736 is shown, the system may comprise a plurality of non-volatile memory devices.

Fig. 16 is a block diagram of a computer initialization system according to another embodiment of the present invention comprising a boot storage device adapter 775 comprising a plurality of volatile logic devices and digital signal processors (or other processors). Here, separate independent dedicated pathways are utilized to load volatile logic devices. Additionally, the outputs of the boot management circuit 1070 are shown for use with other elements of the computer or appliance.

As shown in Figs. 11-16, the non-volatile logic device 736 may be either self-loading or loaded by an external logic device. For example, in Fig. 11, the non-volatile logic device 736 is connected to the volatile logic device 730, which allows the volatile logic device 730 to optionally be self-loading. In Fig. 12, for example, the non-volatile memory device 736 is connected to the DSP 1320 and the volatile logic device 730 has the option of being loaded by the DSP or being self-loaded.

Although illustrative embodiments have been described herein with reference to the accompanying drawings, it is to be understood that the present invention is not limited to those precise embodiments, and that various other changes and modifications may be affected therein by one skilled in the art without departing from the scope or spirit of the invention. All such changes and modifications are intended to be included within the scope of the invention as defined by the appended claims.